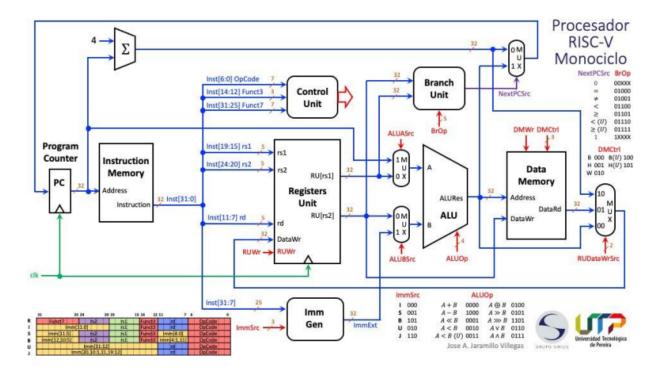
# Laboratotio Arquitectura de Computadoras

### Santiago Ramirez Arenas Docente: José Alfredo Jaramillo Villegas Universidad Tecnológica de Pereira

27 de noviembre de 2021

#### Procesador monociclo.

Este tipo de procesador tiene como principio que cada instrucción debe ser ejecutada en un ciclo del reloj. Cuenta con varios módulos que logran darle la capacidad de procesamiento los cuales están representados en el gráfico.



#### Diseño Proceador Monociclo

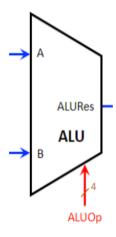
```
'include "instruction_memory.sv"
'include "registers_unit.sv"
3 'include "ALU.sv"
4 'include "data_memory.sv"
5 'include "imm_gen.sv"
6 'include "branch_unit.sv"
7 'include "control_unit.sv"
8 'include "program_counter.sv"
9 'include "adder.sv"
'include "MUX2x1.sv"
'include "MUX3x1.sv"
13 module procesador_monociclo (input CLK);
    wire CLK;
14
    wire [31:0] PC Out;
15
    wire [31:0] adder_Out;
16
    wire [31:0] Instruction;
17
    wire [1:0] RUDataWrSrc;
    wire ALUASrc, ALUBSrc;
19
    wire [31:0] Data1;
20
    wire [31:0] Data2;
21
    wire RUWr;
22
    wire [2:0] ImmSrc;
23
    wire [31:0] ImmExt;
    wire [4:0] BrOp;
25
    wire NextPCSrc;
    wire [31:0] ALURes;
27
    wire [3:0] ALUOp;
    wire [31:0] DataRd;
29
    wire DMWr;
30
    wire [2:0] DMCtrl;
31
    wire [31:0] Out_mux3x1;
32
    wire [31:0] Out_pc_mux;
33
    wire [31:0] ALUMUXResA;
34
    wire [31:0] ALUMUXResB;
35
36
    program_counter pc (
37
             .CLK (CLK,)
38
             .PC_IN (out_pc_mux),
             .PC_Out (PC_Out));
40
41
    adder add (
42
              .adder_In (PC_Out),
43
              .adder_Out (adder_Out));
44
45
    instruction_memory im (
46
               .Addrress (PC_out),
47
               . Instruction (Instruction));
48
49
50
    control_unit cu (
      .OpCode (Instruction [6:0]),
               .Funct3 (Instruction [14:12]),
```

```
.Funct7 (Instruction [31:25]),
                 .ALUASrc (ALUASrc),
54
                  .ALUBSrc (ALUABrc),
55
                 .ALUOp (ALUOp),
56
                 .ImmSrc (ImmSrc),
57
                 .DMWr (DMWr),
58
                  .DMCtrl (DMCtrl),
59
                  .RUDataWrSrc (RUDataWrSrc),
60
                 .RUWr (RUWr),
61
                 .BrOp (BrOp));
62
63
64
     registers_unit ru (
                 .rs1 (Instruction [19:15]),
65
                  .rs2 (Instruction[24:20]),
                 .rd (Instruction [11:7]),
67
                 .DataWr (Out_mux3x1),
                 .RUWr (RUWr),
69
                  .CLK (CLK),
70
                 .Data1 (Data1),
71
                 .Data2 (Data2));
72
74
     imm_gen ig (
                  .Inst (Instruction),
75
                 .ImmSrc (ImmSrc),
76
                  .ImmExt (ImmExt));
77
78
79
     branch_unit bu (
                  .A (Data1),
80
                  .B (Data2),
                  .BrOp (BrOp),
82
                 .NextPCSrc (NextPCSrc));
83
84
     MUX2x1 alu_mux_a (
                 .In1 (Data1),
86
                 .In2 (PC_Out),
87
                 .Src (ALUASrc),
88
                  .Out (ALUMUXResA));
89
90
    MUX2x1 alu_mux_b (
91
                 .In1 (Data2),
92
                  .In2 (ImmExt),
93
                 .Src (ALUBSrc),
94
                 .Out (ALUMUXResB));
95
96
     ALU alu (
97
                  .A(ALUMUXResA),
98
                  .B(ALUMUXResB),
99
                  .ALUOp (ALUOp),
100
                  .ALURes (ALURes));
101
102
     data_memory dm (
103
                  .Address (ALURes),
104
                 .DataWr (Data2),
106
                 .DMWr (DMWr),
```

```
.DMCtrl (DMCtrl),
107
                  .DataRd (DataRd));
108
109
     MUX3x1 mux3to1(
110
                 .In1 (ALURes),
111
                 .In2 (DataRd),
                  .In3 (adder_Out),
                  .RUDataWrSrc (RUDataWeSrc),
114
                  .Out (Out_mux3x1));
116
     MUX2x1 pc_mux(
117
118
       .In1 (adder_Out),
       .In2 (ALURes),
119
       .Src (NextPCSrc),
       .Out (Out_pc_mux));
121
122 endmodule
```

Unidad aritmeticológica (ALU):

Creación de una ALU de dos operandos (A y B tamaño de 32 bits), un ALUOp (tamaño de 4 bits) que asigna una operación y devuelve el resultado (tamaño de 32 bits).



```
1 module ALU (
    input logic signed [31:0] A,
    input logic signed [31:0]B,
    input logic [3:0] ALUOp,
    output logic signed [31:0] ALURes);
5
6
    always @ (*)
7
            begin
              case (ALUOp)
9
                 4'b0000:
10
                      ALURes <= A + B;
                 4'b1000:
12
                      ALURes <= A - B;
                 4'b0001:
14
                      ALURes <= A << B;
15
                 4'b0010:
```

```
ALURes <= A < B;
                  4'b0011:
18
                       ALURes <= $unsigned (A);
19
                  4'b0100:
20
                       ALURes <= A ^ B;
21
                  4'b0101:
22
23
                       ALURes <= A >> B;
                  4'b1101:
24
                       ALURes <= A >>> B;
25
                  4'b0110:
26
                       ALURes <= A | B;
27
28
                  4'b0111:
                       ALURes <= A & B;
29
                   4'b1111:
30
                       ALURes <= A <= B;
31
               endcase
32
             end
33
  endmodule
```

#### • Unidad de registros:

Creación del módulo Unidad de Registros. El cual cuenta con los siguientes registros (cada uno con su respectivo número de bits): el ingreso de la dirección de dos registros(rs1, rs2), dos salidas con la información que fue solicitada (RF[rs1], RF[rs2]), activador de lectura (rd), activador de escritura(RFWr) e información de registro para ingresar(DataWrz).



```
1 module registers_unit (
   input logic
                 [4:0] rs1,
   input logic
                 [4:0] rs2,
   input logic
                 [4:0] rd,
   input logic
                 [31:0] DataWr,
   input logic
                 RUWr,
6
   input logic CLK,
   output logic [31:0] Data1,
8
   output logic [31:0] Data2);
9
   logic [31:0] RU [31:0];
```

```
initial
13
            $readmemb ("registers.txt",RU);
14
    assign Data1 = RU [rs1];
16
    assign Data2 = RU [rs2];
17
18
    always @ (posedge CLK)
19
            begin
20
              if (RUWr == 1 && rd != 5'b0)
                 RU [rd] <= DataWr;</pre>
23
               $monitor ("x20 value: %d", RU [20])
24
  endmodule
```

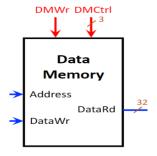
Memoria de datos.

Se define como una matriz en la cual se puede almacenar y leer los datos que van a ser procesados. Este módulo del procesador se encarga de almacenar todos los datos que van a perdurar y, a su vez, recuperar los datos solicitados. El módulo cuenta con los siguientes registros de entrada y salida (cada uno con su respectivo número de bits):

Escritura: La dirección de ingreso para almacenar (Address), el valor que se quierealmacenar (DataWr) y la bandera que activa la escritura (DMWr).

Lectura: Tipo de lectura DMCtrl (Byte, Halfword, Word, Unsigned Byte, Unsigned Halfword), La dirección de ingreso para leer la memoria (Address).

Nota: Address funciona para los casos (lectura y escritura)



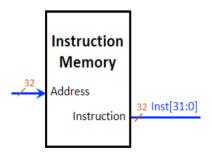
```
module data_memory (
   input logic [31:0] Address,
   input logic signed [31:0] DataWr,
   input logic DMWr,
   input logic [2:0] DMCtrl,
   output logic signed [31:0] DataRd );

integer 1;
   parameter memory_size 2**20;
   logic [7:0] Memory [Memory_size -1: 0];
```

```
11
          //Escritura
12
            always @ (*)
            begin
14
               if (DMWr == 1'b1)
16
                       begin
17
                         case (DMCrel)
                                 3'b000: // Byte
18
                                   Memory [Address] <= DataWr [7:0];</pre>
19
                                 3'b001: // Media palabra
20
                                         begin
21
22
                                   Memory [Address] <= DataWr [7:0];</pre>
                                   Memory [Address + 1] <= DataWr [15:8];</pre>
23
                                         end
24
                                  3'b010: // Palabra
25
                                         begin
26
                                           for (i = 0; 1 < 4; i = i + 1)
27
                               Memory [Address + 1] <= DataWr [i*8 + :8];</pre>
                                         end
2.9
                                endcase
30
                          end
31
                      end
33
             //Lectura
34
                 always @ (*)
35
                 begin
36
                   case (DMCtrl)
37
       3'b000: // Byte
38
           DataRd <= {{24{Memory[Address][7]}}, Memory [Address]};</pre>
39
       3'b001: // Media palabra
40
           DataRd <= {{16{Memory[Address + 1][7]}}, Memory [Address + 1],</pre>
41
      Memory [Address]};
      3'b010: // Palabra
42
           DataRd <= {{Memory[Address + 3], Memory [Address + 2], Memory
43
      [Address + 1], Memory[Address]};
      3'b100://Byte unsigned
44
           DataRd <= {24'b0, Memory [Address]};</pre>
       3'b101 // Media palabra unsigned
46
           DataRd <= {16'b0, Memory [Address + 1], Memory[Address]};</pre>
47
                                     default:
48
                                   DataRd = 31'bx;
49
                          endcase
50
51
                endmodule
```

Memoria de instrucciones.

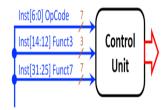
Se puede definir como una matriz en la cual se almacenan y se leen las instrucciones que se van a ejecutar. Este módulo del procesador recibe un Address el cual es la dirección de la instrucción que se quiere recuperar y una salida Instruccion la cual es la instrucción que fue recuperada de la matriz



```
1 module intruction_memory (
    input logic [31:0] Address,
    input logic [31:0] Instruction);
         parameter memory_ize = 2 **20;
6
    logic [7:0] Memory [memory_size - 1: 0];
    initial begin
9
      $readmemh ("instructions.txt", Memory);
11
    always @ (*)
13
      Instruction <={Memory [Address + 3], Memory[Address + 2],</pre>
14
     Memory[Address + 1], Memory [Address]};
15 endmodule
```

Unidad de control

Es la encargada de decodificar la instrucción de entrada (OpCode, Funct3, Funct7) para generar cada una de las señales de control (RUWr, ALUASrc, ALUBSrc, ALUOp, BrOp, DMWr, DMCtrl, RUDataWrSrc e ImmSrc).



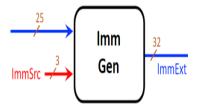
```
1 module control_unit (
    input logic [6:0].OpCode,
    input logic [2:0] Funct3,
    input logic [6:0] Funct7,
    output logic ALUASrc,
    output logic ALUBSrc,
6
7
    output logic [3:0] ALUOp,
    output logic [2:0] ImmSrc,
    output logic DMWr,
9
    output logic [2:0] DMCtrl,
10
    output logic [1:0] RUDataWeSrc,
11
12
    output logic RUWr,
13
    output logic
                   [4:0]BrOp;
14
    always @ (*)
15
            begin
              case (OpCode)
17
                      7'b0110011; //Instrucion tipo R
                      begin
19
                             ALUASrc = 1'b0;
20
                             ALUBSrc = 1'b0;
21
                        ALUOp = {Funct7 [7], Funct3};
                        ImmSrc = 3'bxxx;
23
                        DMWr = 1'b0;
24
                        CMCtrl = 3'bxxx;
25
                        RUDataWrSrc = 2'b00;
26
                        RUWr = 1'b1;
27
                        BrOp = 5' bOOxx;
28
30
                      7'b0010011: // Intruccion tipo I
31
                     begin
32
                             ALUASrc = 1'b0;
                             ALUBSrc = 1'b1;
34
                             if (Funct3 == 3'b101)
                                 ALUOp = {Funct7 [5], Funct3};
36
                             else
37
                               ALUOp = {1'b0, Funct3};
38
                               ImmSrc = 3'b000;
39
                               DMWr = 1'b0;
40
                               DMCtrl = 3'bxxx;
41
                               RUDataWeSrc = 2'b00;
42
                               RUWr = 1'b1;
43
                               BrOp = 5'bOOxx;
44
                      end
45
                      7'b0000011: // Instruccion tipo I load
47
                      begin
                              ALUASrc = 1'b0;
49
                              ALUBSrc = 1'b1;
                              ALUOp = 4'b0000;
                              ImmSrc = 3, b000;
                              DMWr = 1'b0;
53
                              DMCtrl = Funct3;
```

```
RUDataWrSrc = 2'b01;
                                RUWr = 1'b1;
56
                                BrOp = 5'b00xxx;
57
                       end
58
59
                       7'b1100111: // Instruccion tipo I jalr
60
                       begin
61
                                ALUASrc = 1'b0;
62
                                ALUBSrc = 1'b1;
63
                                ALUOp = 4'b0000;
                                ImmSrc = 3'b000;
65
                                DMWr = 1'b0;
                                DMCtrl = 3'bxxx;
67
                                RUDataWrSrc = 2'b10;
                                RUWr = 1'b0;
69
                                BrOp = 5'b1xxxx;
71
                       end
                       7'b1100011: //Instruccion tipo B
73
                       begin
74
                                ALUASrc = 1'b1;
75
                                ALUBSrc = 1'b1;
76
                                ALUOp = 4'b0000;
77
                                ImmSrc = 3'b101;
78
                                DMWr = 1, b0;
79
                                DMCtrl = 3'bxxx;
80
                                RUDataWrSrc = 2'bxx;
81
                                RUWr = 1'b0;
82
                                BrOp = {2'b01, Funct3};
                       end
84
85
                        7'b0100011: //Instruccion tipo S
86
                       begin
                                ALUASrc = 0;
88
                                ALUBSrc = 1;
                                ALUOp = 4'b0000;
90
                                ImmSrc = 3'b001;
91
                                DMWr = 1'b1;
92
                                DMCtrl = Funct3;
93
                                RUDataWrSrc = 2'bxx;
94
                                RUWr = 1'b0;
95
                                BrOp = 5'bOOxx;
96
97
                       7'b1101111: //Instruccion tipo J -Jal
99
                       begin
                                ALUASrc = 1'b1;
101
                                ALUBSrc = 1'b1;
102
                                ALUOp = 4'b0000;
103
                                ImmSrc = 3'b110;
104
                                DMWr = 1'b0;
105
                                DMCtrl = 3'bxxx;
106
                                RUDataWrSrc = 2'b10;
107
108
                                RUWr = 1'b1;
```

```
BrOp = 5'b1xxxx;
109
                        end
110
111
                        7'b0110111: //Instruccion tipo U -lui
112
                        begin
113
                                 ALUASrc = 1'bx;
114
                                 ALUBSrc = 1'b1;
                                 ALUOp = 4'b1111;
116
                                 ImmSrc = 3'b010;
117
                                 DMWr = 1'b0;
118
                                 DMCtrl = 3'bxxx;
119
                                 RUDataWrSrc = 2'b00;
120
                                 RUWr = 1'b1;
121
                                 BrOp = 5'b100xxx;
                        end
124
                        7'b0110111: //Instruccion tipo U -auipc
                        begin
126
                                 ALUASrc = 1'b1;
127
                                 ALUBSrc = 1'b1;
128
                                 ALUOp = 4'b0000;
                                 ImmSrc = 3'b010;
130
                                 DMWr = 1, b0;
131
                                 DMCtrl = 3'bxxx;
132
                                 RUDataWrSrc = 2'b00;
133
                                 RUWr = 1'b1;
                                 BrOp = 5'b100xxx;
135
                        end
136
137
                endcase
138
     endmodule
139
```

Generador de inmediatos

El generador de inmediatos es el encargado de ensamblar el inmediato decodificado en la instrucción de entrada según el tipo de esta



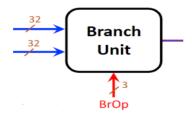
```
module imm_gen(
input logic [31:0] Inst,
input logic [2:0] ImmSrc,
output logic [31:0] ImmExt);

always @ (*)
begin
case (ImmSrc)
```

```
3'b000: //Intruccion tipo I
                ImmExt = \{\{20\{Inst\{31\}\}\}, Inst \{31:20\}\};
            3'b001: //Intruccion tipo S
11
              ImmExt = \{\{20\{Inst\{31\}\}\}, Inst\{31:25\}, Inst\{11:7\}\};
            3'b101: //Intruccion tipo B
13
               ImmExt = {{19{Inst{31}}}, Inst {31}, Inst{7},Inst {30:25},Inst
14
      {11:8}, 1'b0};
            3'b010: //Intruccion tipo U
               ImmExt = {{Inst{Inst{31:12}}}, 12'b0};
16
            3'b110: //Intruccion tipo J
17
               ImmExt = \{\{19\{Inst\{31\}\}\}, Inst\{31\}, Inst\{19:12\}, Inst\}\}
18
      {20}, Inst {30:21}, 1'b0};
                 default;
19
                 ImmExt = 31'bx;
20
         endcase
21
22
       end
23 endmodule
```

#### Branch unit

Esta se encarga de determinar si la instrucción de entrada ordena un salto en la dirección de la memoria de instrucciones



```
1 module branch_unit(
    input logic signed
                          [31:0] A,//Es asigando de RU [rs1]
                          [31:0] B, //Es asignado de RU [rs2]
    input logic signed
    input logic [4:0] BrOp,
4
    output logic NextPCSrc );
6
    alwats 0 (*)
7
          begin
8
            if (BrOp[4])
                   NextPCSrc = 1;
                   else
                     if (BrOp [3])
12
                       case (Br0[2:0])
13
                         3'b000; //BEQ
14
                           NextPCSrc = (A == B);
                         3'b000; //BNE
16
                           NextPCSrc = (A != B);
17
                         3'b100; //BLT
                           NextPCSrc = (A < B);</pre>
19
                         3'b101; //BGE
20
                           NextPCSrc = (A >= B);
21
                         3'b110; //BLTU
```

```
NextPCSrc = ($unsigned (A) < $unsigned (B));</pre>
                         3'b111; //BGEU
2.4
                         NextPCSrc = ($unsigned (A) >= $unsigned (B));
                         default:
26
                                 NextPCSrc = 1'bx;
27
                       endcase
28
                  else
                         NextPCSrc = 0;
30
          end
31
    endmodule
```

### Program counter

#### Adder

```
module adder (
input [31:0] adder_In,
output [31:0] adder_Out );

assign adder_Out = adder_In + 4;
endmodule
```

#### MUX 2x1

```
1 module MUX2x1 (
   input logic [31:0] In1,
   input logic [31:0] In2,
   input logic Src,
    output logic [31:0]Out);
5
    always @ (*)
7
           begin
             case (Src)
9
                     1'b0:
                             Out <= In1;
                     1'b1:
                             Out <= In2;
13
             endcase
14
16 endmodule
```

#### MUX 3x1

```
module MUX3x1 (
input logic [31:0] In1,
input logic [31:0] In2,
input logic [31:0] In3,
```

```
input logic [1:0] RUDataWrSrc,
    output logic [31:0]Out);
    always @ (*)
8
            begin
              case (RUDataWeSrc)
11
                      2'b00:
                             Out <= In1;
12
                      2'b01:
13
                             Out <= In2;
14
                      2'b10:
                             Out <= In3;
17
              endcase
            end
19 endmodule
```

#### Testbench

```
module testbench;
         logic CLK;
    procesador_monociclo pm (CLK)
           always
                   #5
                   CLK = ~CLK;
8
          initial
9
                 begin
10
                    $dumpfile ("dump.vcd");
11
                    $dumpvars();
12
                         CLK=0;
13
                         #850
14
                         $finish;
15
                  end
17 endmodule
```

#### Instructions

```
1 FF010113
2 00812623
3 00912423
4 01212223
5 01312023
6 00050413
7 00000913
8 00000993
9 00000993
10 0099D863
11 00890933
12 00198993
13 FE000AE3
14 00090513
15 00C12403
16 00812483
17 00412903
18 00012983
```

```
19 01010113
20 00008067
  00040513
22 OFC000EF
23 00050493
24 FF010113
  00812623
26 00912423
  01212223
  00112023
  00050413
30 00100913
31 00140413
32 00100493
  0084DE63
34 00090513
35 00048593
36 DD1FF0EF
37 00050913
38 00148493
39 FE0004E3
40 00090513
41 00C12403
42 00812483
43 00412903
  00012083
45 00012083
46 01010113
47 00008067
```

#### Registers

#### Resultado

```
#KERNEL: x20 value 0
#KERNEL: x20 value 72
RUNTIME: Info: RUNTIME_0068 testbemch.sv (16): $finish calle.
#KERNEL: Time: 1 us, Iteration: 0, Instance: / testbench, Process:
    @INITIAL#19_1@.
#KERNEL: stopped at time: 1 us
#VSIM: Simulation has finished. There are no more test vectors to simulate.
#VSIM: Simulation has finished
```

#### Simulación



## Instrucciones RV32I

	Tip	o R			Tipo I			Tipo S	В
	OpCode	Funct3	Funct7		OpCode	Funct3		OpCode	Funct3
add	0110011	000	0000000	addi	0010011	000	beq	1100011	000
sub	0110011	000	0100000	slli	0010011	001	bne	1100011	001
sll	0110011	001	0000000	slti	0010011	010	blt	1100011	100
slt	0110011	010	0000000	sltiu	0010011	011	bge	1100011	101
sltu	0110011	011	0000000	xori	0010011	100	bltu	1100011	110
xor	0110011	100	0000000	srli	0010011	101	bgeu	1100011	111
srl	0110011	101	0000000	srai	0010011	101	2900		
sra	0110011	101	0100000	ori	0010011	110	sb	0100011	000
or	0110011	110	0000000	andi	0010011	111	sh	0100011	001
and	0110011	111	0000000				sw	0100011	010
				1b	0000011	000	54	0100011	010
	Tipo	l U J		1h	0000011	001			
_				lw	0000011	010			
		OpCode		1bu	0000011	100			
		101111		lhu	0000011	101			
	lui 0	110111		jalr	1100111	000			
	auipc 0	010111		Jair	1100111	000			Universidad Tecnológ
							Jose A. Jarami	lo Villegas GRUPO	SIBILIS de Pereira

Crear un código en C ++, llevar a ensamblador y por último a lenguaje de máquina. El cual será utilizado como las instrucciones de prueba para demostrar el correcto funcionamiento.

```
Funcion en C++
3 o Funcion en C++
5 int multi(int x, int y)
7 int Acc = 0;
8 for(int i = 0; i < y; i++) Acc += x;</pre>
9 return Acc;
10 }
11
int factorial(int num)
14 int fact = 1;
15 int num += 1;
16 for(int i = 1;i < num;i++) fact = multi(fact,i); return fact;</pre>
17 }
18
19 o Funcion en lenguaje ensamblador
21 Prototipo de la funcion multi:
                 x8 x11
                                     x9
22 x10
         X
                              У
23 Acc
          x18
       x19 multi(x,y)
                             x10
26 Prototipo de la funcion factorial: x10
                                                  num
                                                           x8
             x18 factorial(num)
                                      x10
28 fact
```

```
29
30 Llamado a la funci n factorial: f = factorial(num)
31 num x8
32 f x9
```

		Lenguaje ensamblador	
Address	Label	Instruction	Description
0x04200	multi:	addi sp,sp,-16	Reservamos espacio en la pila
0x04204		sw x8,12(sp)	Guardamos en los espacios
0x04208		sw x9,8(sp)	reservados anteriormente
0x0420C		sw x18,4(sp)	
0x04210		sw x19,0(sp)	
0x04214		addi x8,x10,0	Movemos los argumentos
0x04218		addi x9,x11,0	
0x0421C		addi x18,x0,0	Acc = 0
0x04220		addi x19,x0,0	i = 0
0x04224	for0:	bge x19,x9,endfor0	Cuando (i >= y) salir del for
0x04228		add x18,x18,x8	Acc += x
0x0422C		addi x19,x19,1	i++
0x04230		beq x0,x0,for0	
0x04234	endfor0:	addi x10,x18,0	return Acc
0x04238		lw x8,12(sp)	Recuperar de la pila los
0x0423C		lw x9,8(sp)	valores en los que recibí los
0x04240		lw x18,4(sp)	registros.
0x04244		lw x19,0(sp)	
0x04248		addi sp,sp,16	Liberar el espacio
0.10.11.12.10		addi sp,sp,10	reservado
0x0424C		jalr x0,0(x1)	Retorno
•••			
0x04300		addi x10,x8,0	
0x04304		jal x1,factorial	
0x04308		addi x9,x10,0	
0x04400	factorial:	addi sp,sp,-16	Reservar espacio en la pila.
0x04404		sw x8,12(sp)	Guardar en el espacio
0x04408		sw x9,8(sp)	reservado en la pila los
0x0440C		sw x18,4(sp)	registros.
0x04410		sw x1,0(sp)	
0x04414		addi x8,x10,0	Mover los argumentos
0x04418		addi x18,x0,1	fact = 1
0x0441C		addi x8,x8,1	num += 1
0x04420		addi x9,x0,1	i = 1
0x04424	for0:	bge x9,x8,endfor0	Si (i >= num) salir del for
0x04428	20201	addi x10,x18,0	fact = multi(fact,i)
0x0442C		addi x11,x9,0	
0x04430		jal x1,multi	
0x04434		addi x18,x10,0	
0x04438		addi x9,x9,1	i++
0x0443C		beq x0,x0,for0	TTT
0X0443C		bed vo'vo'vo	

0x04440	endfor0:	addi x10,x18,0	return fact
0x04444		lw x8,12(sp)	Recuperar de la pila los
0x04448		lw x9,8(sp)	valores en los que recibí los
0x0444C		lw x18,4(sp)	registros.
0x04450		lw x1,0(sp)	
0x04454		addi sp,sp,16	Liberar el espacio
			reservado
0x04458		jalr x0,0(x1)	Retorno

												: <u></u>			ا ماما	N 4 6 00		F	a: 4 .a	N 4 l a										
											onve	rsior	n len		di sp,			Fund	cion	IVIUI	.1									
				Im	ım [1	1.01							Rs1	Aut	ıı sp,		unct				rd						pcod	40		
1	2	3	4		1111 <u>[ 1</u> 5   7	8	9	10	11	12	13	14	15	16	17	г 18	29	.5 20	21	22	23	24	25	26	27	28	29	30 I	31	32
1	1	1	1		$\frac{3}{1}$ $\frac{7}{1}$	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1
F		_		F			0				1				0				1				1	Ü	_	_	3			
													Hex	adec	imal	0Xff	0101	113												
															x8, 1															
	I	mm	[11:	:5]				Rs2				-	Rs1				unct	:3		Im	m [4	:0]				0	рсос	de		
1	2	3			5 7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0 (	0 0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0	1	0	0	0	1	1
0				0			8				1				2				6				2				3			
													Hexa	adeci	imal	0x00	812	623												
														Sv	v x9,	8 (sp	)		•					•						
		mm						Rs2					Rs1	Т			unct				m [4						pcod			
1	2	3			5 7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0			0   0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	1
0				0			9				1				2		040	400	4				2				3			
													неха					423												
Hexadecimal 0x00912423   Sw x18, 4 (sp)   Imm [11:5]   Rs2   Rs1   Funct3   Imm [4:0]																				1 -										
1	2	mm 3			5 7	8	9	Rs2	11	12	13	14	Rs1 15	16	17	18	29	20	21	22	m [4	:U] 24	25	26	27	28	29	30 S	31	32
0	0	_			0 0		0	0	1	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1
0	U			1	<u> </u>	_	2				1			_	2		_		2		_		2	Ü	-	Ü	3	Ü	-	-
													Hexa	deci		0x01	212	223												
															x19,															
	I	mm	[11:	:5]				Rs2				-	Rs1				unct	:3		Im	m [4	:0]				0	рсос	de		
1	2	3	4	5 6	5 7	8	9	10	11	12	13	14	15	16	17	18	20	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0 (	0 0	1	0	0	1	1	_					10	29	20	21	22				20	21	20		50	31	1
9				1			1 ~		-		0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	
							3				1	0	0	1	2						0	0								_
												0				0	1	0	0		0	0	0				0			_
												0	Hex	adec Adc	2 imal li x8,	0 0x13 x10,	1 3120 0	0	0			0	0			0	3	0		
	-				ım [1	_					1		Hex Rs1	adec Adc	2 cimal li x8,	0 0x13 x10,	1 3120 0 unct	0)23	0	0	rd		2	0	1	0	0 3 pcod	0 de	1	
1	2	3		5 (	5 7	8	9	10		12	13	14	Hex Rs1	adec Adc	2 cimal li x8, 17	0 0x13 x10, F 18	3120 0 unct 29	0 23 3 20	0 0	22	rd 23	24	2 25	26	27	O 28	0 3 pcoc	0 de 30	31	32
0	2	3	0	5 ( 0 (		8	9	10	11 0		13 0		Hex Rs1	adec Adc	2 di x8,	0 0x13 x10,	1 3120 0 unct	0)23	0 0 21 0	0	rd		2 2 25 0	0	1	0	0 3 pcoo	0 de	1	
			0	5 (	5 7	8	9			12	13	14	Rs1	Ado	imal li x8,	0 0x13 x10, F 18 0	3120 0 unct 29	0 23 3 20 0	0 0	22	rd 23	24	2 25	26	27	O 28	0 3 pcoc	0 de 30	31	32
0			0	5 ( 0 (	5 7	8	9			12	13 0	14	Hex Rs1	Add 16 1	imal 2 di x8,	0 0x13 x10, F 18 0	3120 0 unct 29 0	0 23 3 20 0	0 0 21 0	22	rd 23	24	2 2 25 0	26	27	O 28	0 3 pcoo	0 de 30	31	32
0			0	5 (c) (d) (d) (d) (d) (d) (d) (d) (d) (d) (d	5 7 0 0	8	9			12	13 0	14	Rs1 15 0	Add 16 1	imal li x8,	0 x10, F 18 0 0x00	1 3120 0 unct 29 0	0 23 3 20 0 413	0 0 21 0	22	rd 23 0	24	2 2 25 0	26	27	0 0 28 1	0 3 pcoo	0 de 30 0	31	32
0		0	0	5 ( 0 ( 0	7 0 0	1:5]	9 0 0	0	0	12 0	13 0 5	14 1	Rs1 15 0 Hexa	adeci Ado 16 1	2 di x8, 17 0 0 imal di x1	0 x10, F 18 0 0x00 8,x0,	3120 0 unct 29 0 0504 0	0 23 3 20 0 413	0 0 21 0 4	22 1	rd 23 0	24 0	2 25 0 1	26 0	27 0	O O 28 1	0 3 pcoo	0 de 30 0	31 1	32
0	0		4	5 (c) 0 (d) 0 (d) 5 (d) 6 (d)	5 7 0 0	1:5] 8	9			12	13 0	14	Rs1 15 0	adeci Ado 16 1	imal 2 di x8,	0 x10, F 18 0 0x00	1 3120 0 unct 29 0	0 23 3 20 0 413	0 0 21 0	22	rd 23 0	24	2 2 25 0	26	27	0 0 28 1	0 3 pcoo	0 de 30 0	31	32
0	2	3	4 0	5 (c) 0 (d) 0 (d) 5 (d) 6 (d)	7 0 0 nm [1 5 7	1:5] 8	9 0 0	10	11	12 0	13 0 5	14 1	Rs1 15 0 Hexa	adeci Ado 16 1	2 di x8, 17 0 0 imal di x1	0 x10, F 18 0 0x00 8,x0, F	3120 0 unct 29 0 0504 0 unct	0 23 3 20 0 413	0 0 21 0 4	22 1	rd 23 0 m [4 23	24 0 :0] 24	25 0 1	26 0	27 0	O 28 1 O 28	0 3 pcoo 29 0 3	0 de 30 0	31 1	32 1
0 0 1 0	2	3	4 0	5 (c) 0 (d) 5 (d) 6 (d)	7 0 0 nm [1 5 7	1:5] 8	9 0 0	10	11	12 0	13 0 5	14 1	Rs1 15 0 Hexa Rs1 15 0	adeci Adci 16 1 Adci 16 1	2 simal di x8, 0 0 imal di x1	0 0x11 x10, F 18 0 0x00 8,x0, F 18	1 3120 0 unct 29 0 0 0 0 0 0 0	0 23 20 0 413 20 0	0 0 21 0 4 21 0	22 1	rd 23 0 m [4 23	24 0 :0] 24	25 0 1 25 1	26 0	27 0	O 28 1 O 28	0 3 pcoo 29 0 3	0 de 30 0	31 1	32 1
0 0 1 0	2	3	4 0	5 (c) 0 (d) 5 (d) 6 (d)	7 0 0 nm [1 5 7	1:5] 8	9 0 0	10	11	12 0	13 0 5	14 1	Rs1 15 0 Hexa Rs1 15 0	Add  16 1  Addeci Add  16 1  16 1  adeci Add  16 1	2 simal di x8, 0 0 imal di x1	0 0x10, F 18 0 0x00 8,x0, F 18 0	1 3120 0 unct 29 0 0504 0 unct 29 0	0 23 20 0 413 20 0	0 0 21 0 4 21 0	22 1	rd 23 0 m [4 23	24 0 :0] 24	25 0 1 25 1	26 0	27 0	O 28 1 O 28	0 3 pcoo 29 0 3	0 de 30 0	31 1	32 1
0 0 1 0	2	3	4 0	5 (c) (0 (c)	7 0 0 nm [1 5 7	1:5] 8	9 0 0	10	11	12 0	13 0 5	14 1	Rs1 15 0 Hexa Rs1 15 0	Add  16 1  Addeci Add  16 1  16 1  adeci Add  16 1	2 imal li x8,  17 0 0 imal di x1  17 1 8	0x11x10, x10, F 18 0 0x00 8,x0, F 18 0	1 3120 0 unct 29 0 0504 0 unct 29 0	0 0 23 20 0 413 3 20 0	0 0 21 0 4 21 0	22	rd 23 0 m [4 23	24 0 :0] 24 0	25 0 1 25 1	26 0	27 0	O 28 1 O 28 1	0 3 pcoo 29 0 3	0 30 0	31 1	32 1
1 0 0	2	3	4 0	5 (c) (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	7 0 0 0 0 0 0 0 0 0 0 0 0	1:5] 8 0	9 0 0	10	11 0	12 0	13 0 5	14 1	Rs1 15 0 Hexa	Add  16 1  Addeci Add  16 1  16 1  adeci Add  16 1	2 imal li x8,  17 0 0 imal di x1  17 1 8	0x11x10, x10, F 18 0 0x00 8,x0, F 18 0	1 3120 0 unct 29 0 0 0 0 unct 29 0	0 0 23 20 0 413 3 20 0	0 0 21 0 4 21 0	22	rd 23 0 m [4 23 0	24 0 :0] 24 0	25 0 1 25 1	26 0	27 0	O 28 1 O 28 1	0 3 29 0 3 3 pcoo 29 0 3	0 de 30 0 de 30 de de 30 de	31 1	32 1
0 0 1 0 0	2 0	3 0	4 0	5 (0 (0 (0 (0 (0 (0 (0 (0 (0 (0 (0 (0 (0	mm [1 5 7 0 0	1:5] 8 0	9 0 0	10 0	11 0	12 0	1 13 0 5 13 0 5	14 1	Rs1 15 0 Hexa Rs1 15 15 Rs1 Rs1	adec Add  16 1  Add  16 1  Add  Add  Add	2 imal di x1 8 5 imal di x1 9 imal di x1 9 5 imal di x1 9 imal di x1 9 5 imal di x1 9 imal di x1 9 imal di x1 9 imal di x1 9 i	0 0x11 x10, F 18 0 0x00 F 18 0 0x00 0x00 F 18 0	1 3120 0 unct 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 23 20 0 413 20 0	21 0 4	22	rd 23 0 m [4 23 0	24 0 :0] 24 0	25 0 1 25 1 9	26 0	27 0	O 28 1 O O O O O O O O O O O O O O O O O O	0 3 pcoo 29 0 3	0 de 30 0 de 30 de de 30 de	31 1 31 1	32 1 32 1

														المدر	doc:	mal	<u></u>	0000	012												
	0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 0 1																														
1						7	8	9			12	13	14		16	17				21				25	26	27				31	32
0																	1														
0												9								8				6							
														Hexa	deci	mal (	0x00	099D	863												
															Add	x18	,x18	3,x8													
	I	mm	[11	<u>.:5]</u>					Rs2					Rs1			F	unct	:3		lm	m [4	:0]				0	рсос	de		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	1	1	0	0	1	1
0				0				8				9				0				9				3				3			
																		0890	933												
						[4.4	1.01								Add	x19					1	[ ^	.01						J.,		
															30	31	32														
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	1	0	0	1	1	0	0	1	0	0	1	1
0	J	J		0	J			1			-	9			-	8				9	1	,	-	9	<u> </u>	-		3	-5-		
Ť														Hexa	deci		0x0(	01989	993												
																x0, x															
	lm	m [:	12,1	0;1	5]				Rs2					Rs1				unct	:3		Imm	[4:0	,11]				0	рсос	de		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	0	0	0	1	1
F				E				0				0				0				Α				Ε				3			
														Hexa				E000	AE3												
															Add	i x10									1						
1	2	3	4	5 5	mm 6	[11 7	L:0] 8	9	10	11	12	13	14	Rs1 15	16	17	18	unct 29	20	21	22	rd 23	24	25	26	27	28	<b>pco</b> 0		31	32
0	0	0	0	0		0		0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	1
0	U	U	Ū	0	U			0	U	U	O	_	٥		_	0				5	_	U	_	2	Ü	Ū	_	3	Ü	-	
Ť															deci		0x0(	0090	513												
																x8, 1															
				I	mm	[11	L:0]							Rs1				unct	:3			rd					0	рсос	de		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18		20	21	22	23	24	25	26	27	28	29		31	32
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	1
0				0				С				1				2	•	2017	100	4				0				3			
-														Hexa				)C12	403												
						[4.4	1.01							D : 1	lv	/ x9,8		•				1							l <u>-</u>		
1	2	3	4	5 5	mm 6	[11 7	L:0 <u>]</u> 8	9	10	11	12	13	14	Rs1 15	16	17		unct 29	20	21	22	rd 23	24	25	26	27	28	pcod	1e 30	31	32
0	0	0	0	0	0	0		1	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1	1
0	J	J		0	J			8			3	1			-	2		_		4		,		8	- <del>-</del> -	-	-	3	~	-	
														Hexa	deci		0x00	08124	483												$\dashv$
																x18,															=
	ı	mm	[11	L:0]					Rs2					Rs1				unct	:3			rd					0	pcod	le	-	
1	2	3	4	5	6	7	8	9	10		12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1
0				0				4				1				2				9				0				3			

															.1	14	2 00	142	202												
														неха		mal (			903												
						[4.4	.01							D-1	IW	x19,															
1	2	3	4	5 5	mm 6	7		9	10	11	12	13	14	Rs1 15	16	17	18	unct 29	20	21	22	rd 23	24	25	26	27	28	29	ae 30	31	32
0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	1	1	0	0	0	0	0	1	1
0	U	U	U	0	U	U	U	0	U	U	U	1	U	U		2	U		0	9	U	U	Т	8	U	U	U	3	U	1	
-				U				U						Lova	ndoci	mal (	OvOr	1012	กดว	)				0				3			
														пеха		li sp,			303												
					mm	[11	· <b>∩</b> 1							Rs1	Auc	ıı sp,		unct				rd						рсос	40		
1	2	3	4	5	6	7	.UJ 8	9	10	11	12	13	14	15	16	17	<u>г</u> 18	29	.5 20	21	22	23	24	25	26	27	28	29	30 I	31	32
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1
0			-	1			-	0		U	•	1				0				1				1	Ü	Ū	-	3	Ü	-	
<u> </u>														Heya	adeci	mal (	በአበኅ	010	113												
														TICAC		r x0,			113												
				1	mm	[11	:01							Rs1	Juil	. ٨٥,		<del>+)</del> unct	·3			rd						рсос	1e		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1
0				0				0				0				8				0				6				7			
														Hexa	adeci	mal	0x00	0008	067												
																li x1(															
				1	mm	[11	:01							Rs1				unct	3			rd					0	рсос	de		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	1
0				0				0				4				0				5				1				3			
														Hexa	adeci	mal	0x0(	040	513												
															Jal	1, fa	ctor	ial													
							I	mm	[20,	,10: 1	l, 11	,19:1	L2]									rd					0	рсос	de		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1
0				F				С				0				0				0				Ε				F			
														Hexa		mal			0EF												
																li x9,															
						[11							ı	Rs1				unct			1	rd	ı	ı				pcod			
1	2	3	4	5	6	7	8	9	10		12	13	14	15	16	17	18		20	21	22	23	24	25	26	27		29		31	32
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	1
0				0				0				5				0				4				9				3			
-														Hex		imal			193												
<u> </u>															Add	i sp,						•	•								
_		11:5	-	_			_		4.0		4.0	4.0		Rs1	1.0			unct		2.5		m [4		25	2.0			pcod		24	
1	2	3 1	4	5	6	7	8	9	10 0	11	12	13	14	15	16	17	18 0		20	21	22	23	24	25	26	27	28	29		31	32
1	I	I	1	1	1	1	1	0	U	U	0	1	0	0	I	0	U	0	0	0	0	U	1	0	0	0	1	0	0	1	1
F				F				0				1		Цан	- d :	0	٥,,,,,	010	112	1				1				3			
														пеха		mal			113												
		100:00	[11	1					D-2					Dc1	5W	x8, 1					1	po [4	۱۵۱					000	40		
1	2	mm 3	4	.:5] 5	6	7	8	9	Rs2	11	12	13	14	Rs1 15	16	17	18	unct 29	20	21	1m 22	m [4 23	:U] 24	25	26	27	28	<b>pco</b> 0		31	32
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0	1	0	0	0		1
0	U	U	U	0	J	J	J	8	0	U	0	1	U	J	-	2	J		U	6		1	U	2	0	_	O	3	-0	T	Ŧ
U				U				٥				Т								O								Э			

														Have	ماد د:	m - 1	0,400	1012	622												
														неха	deci				623												
		mm	[11	·c1					Rs2					Rs1	3W	x9,		unct	· ɔ		Im	m [4	·01					рсос	40		
1	2	3	4	ວ <sub>]</sub> 5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	.5 20	21	22	23	24	25	26	27	28	29	30 30	31	32
0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	1
0			_	0				9				1				2				4				2	Ť			3		_	
														Hexa	deci		0x00	912	423	l				I							
																x18,															
	I	mm	[11	.:5]					Rs2					Rs1			-	unct	:3		lm	m [4	:0]				0	рсос	de		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1
0				1				2				1				2				2				2				3			
														Hexa	adeci	mal	0x01	212	223												
															W2	2 x1,	0 (sp	o)													
<u> </u>		mm					-1		Rs2					Rs1	,			unct				m [4	-	Т				pcod			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1
0				0				1				1				2	0.00			0				2				3			
														неха					023												
	Hexadecimal 0x00112023  Addi x8, x10, 0  Imm [11:0] Rs1 Funct3																							d z							
1	2	3	4	5	mm 6	7	:U] 8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	rd 23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1
0			Ū	0			Ū	0			U	5	_		_	0				4	_			1	Ū	J	_	3	Ū	_	_
			<u> </u>											Hexa	deci		0x00	050	413												
																li x18															
				ı	mm	[11	:0]							Rs1				unct	:3			rd					0	рсос	de		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1	1
0				0				1				0				0				9				1				3			
														Hexa	deci				913												
																8x ib				1					1						
L.,						[11				1 .			1 .	Rs1				unct				rd	I _	I _				pcod			
0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	0	0	0	U	1	0 4	1	0	0	0	U	0	0	<u>0</u> 4	1	0	0	0 1	0	0	1	3	U	1	1
U				U								4		Hova	adeci		กงกก	11/10	/112	4								3			
														пехс		di x9			413												
				1	mm	[11	.U1							Rs1	Au	ui X9		unct	. a			rd						рсос	1 <sub>0</sub>		
1	2	3	4	5	6	7	Uj 8	9	10	11	12	13	14	15	16	17	18	29	.5 20	21	22	23	24	25	26	27	28	29		31	32
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	1
0				0				1				0								4				9				3			
			ı											Hexa	deci	mal	0x00	100	493												
														B	ge x9	, x8,	end	for 0	)												
	ı	mm	[11	:5]					Rs2					Rs1	_			unct			lm	m [4	:0]				0	рсос	de		
4		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3		5	0		0	9	10	11	12	13																			
0	0	0	0	0	0	0	0	1 8	0	0	0	0	1	0	0	<b>1</b> D	1	0	1	1 E	1	1	0	<b>0</b>	1	1	0	0	0	1	1

														Hexa	ideci	mal (	0x00	84D	E63												
																i x10															
																	:3			rd					О	рсос	de				
1	2	3	4	5		_						13	14	15		17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0		0	0	0	0	0	0	0	_	0	0	1		0	0	0	0	1	0	1	0	0	0	1	0	0	1	1
0				0				0				9								5				1				3			
														Hexa					513												
						[44	01					l		D . 1	Add	li x11			2			1							.1 .		
1	2	3	4	5 5	mm 6	[11 7	:U] 8	9	10	11	12	13	14	Rs1 15	16	17	18	unct 29	20	21	22	rd 23	24	25	26	27	28	29	ae 30	31	32
0	0	0	0	0			0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1	1	0	0	1	0	0	1	1
0				0				0				4	_		Ū	8				5	_		_	9	J	Ū	_	3	Ū	_	_
		Hexadecimal 0x00048593																													
		Jal x, multi																													
		Imm [11:5] rd																		0	рсос	de									
1	2	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 29 20 21 22 23 24															24	25	26	27	28	29	30	31	32						
1	1	1 0 1 1 1 0 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0														0	1	1	1	0	1	1	1								
D		D 1 F F 0															Ε				F										
		D																													
	Addi x18, x10, 0																														
		1 _ 1			mm			_		1			T	Rs1				unct				rd				1		pcod			
1	2	3	4	5	6 <b>0</b>	7	8	9 <b>0</b>	10 <b>0</b>	0	12 0	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29 0	30	31	32
0	0	U	0	0	U	U	U	0	U	U	U	0	1	0	1	0	U	0	0	1	0	U	1	0	0	0	1	3	0	1	1
																				ıu				1 1							
-				U				U				5		Hexa	ndeci	0 mal (	0x00	0509	913	9				1				3			
				0				U				5		Hexa		mal (			913	9				1				3			
					mm	[11	:01	<u> </u>				5		Hexa			, x9,			9		rd		1			0		de		
1	2	3	4			7	:0]	9	10	11	12	13	14			mal (	, x9,	1		21	22	rd 23	24	25	26	27	O 28	)pco( 29		31	32
	2	3 0	4 0	1	6	7			10	11 0	12 1	13 0		Rs1	Adı	mal ( di x9,	, x9, F	1 unct	3		22		24	25 1	26 0	27		)pco( 29 0		31	32
1				J 5	6	7	8	9				13	14	Rs1 15 0	16 0	17 1 8	, x9, F 18	1 unct 29 0	20	21		23		25			28	)pco(	30		
1 0				5	6	7	8	9				13 0	14	Rs1 15 0	Add	17 1 8	, x9, F 18 0	1 unct 29 0	20	21		23		25 1			28	)pco( 29 0	30		
1 0	0	0	0	5 0 0	6	7	8	9 0 1	0			13 0	14	Rs1 15 0	16 0 adeci	17 1 8	0 0 0 0 0 0,foi	1 unct 29 0 1484	3 20 0	21	1	0	0	25 1 9			28	29 0 3	30		
1 0 0	0 Ir	nm	[11:	0 0 0	6 0	7 0	8	9 0 1	0 Rs2	0	1	13 0 4	14	Rs1 15 0 Hexa	16 0 adeci	17 1 8 (mal (	0x00 (0,foi	1 unct 29 0 1484 10 unct	3 20 0 493	21 0 4	1 Imm	23 0	0,11]	25 1 9	0	0	28 1	29 0 3	30 0	1	1
1 0 0	0   Ir   2	mm 3	[11:4	0 0 0 :0]	6 0	7 0	8 0	9 0 1	0 Rs2	11	12	13 0 4	14 1	Rs1 0 Hexa Rs1	Ado 16 0 adeci Beq	17 1 8 mal ( x0, x	0x00 (0,fo)	1 unct 29 0 1484 0 unct 29	3 20 0 493 3 20	21 0 4	1 Imm 22	23 0 1 [4:0 23	0,11]	25 1 9	26	27	28 1 C 28	29 0 3	30 0	31	32
1 0 0	0 Ir	nm	[11:	5 0 0 :0]	6 0	7 0	8 0	9 0 1	0 Rs2	0	1	13 0 4 13 0	14	Rs1 15 0 Hexa	16 0 adeci	mal (di x9,	0x00 (0,foi	1 unct 29 0 1484 10 unct	3 20 0 493	21 0 4	1 Imm	23 0	0,11]	25 1 9	0	0	28 1	0 29 0 3	30 0	1	1
1 0 0	0   Ir   2	mm 3	[11:4	0 0 0 :0]	6 0	7 0	8 0	9 0 1	0 Rs2	11	12	13 0 4	14 1 14 0	Rs1 15 0 Hexa	16 0 adeci Beq 16	17 1 8 mal ( x0, x	0x00 0x00 0x0,foi	1 unct 29 0 1484 0 unct 29	3 20 0 493 3 20 0	21 0 4	1 Imm 22	23 0 1 [4:0 23	0,11]	25 1 9	26	27	28 1 C 28	29 0 3	30 0	31	32
1 0 0	0   Ir   2	mm 3	[11:4	5 0 0 :0]	6 0	7 0	8 0	9 0 1	0 Rs2	11	12	13 0 4 13 0	14 1 14 0	Rs1 15 0 Hexa	16 0 adeci Beq 16 0	17 1 8 mal (x0, x	0x00 (0,fo) 18 0 0x00 (0,fo)	1 29 0 1484 0 0 unct 29 0	3 20 0 493 3 20 0	21 0 4	1 Imm 22	23 0 1 [4:0 23	0,11]	25 1 9	26	27	28 1 C 28	0 29 0 3	30 0	31	32
1 0 0	0   Ir   2	mm 3	[11:4	0 0 0 5 1 E	6 0	7 0 7 7 1	8 0	9 0 1	0 Rs2	11	12	13 0 4 13 0	14 1 14 0	Rs1 15 0 Hexa	16 0 adeci Beq 16 0	mal (di x9, 17 1 8 mal (x0, x 17 0 0 mal (mal (di x0))	0x00 0x00 0x0,foi 18 0 0XFE x18,	1 29 0 1484 0 0 unct 29 0	3 20 0 493 3 20 0	21 0 4	1 Imm 22	23 0 1 [4:0 23	0,11]	25 1 9	26	27	28 1 C 28 0	29 0 3 3 0 0 29 0 3	30 0 de 30 0	31	32
1 0 0 1 1 F	0   Ir   2   1	mm 3 1	0 [11: 4 1	I 5 0 0 0 5 1 E	6 0 1 mm 6	7 0 7 1 1 [11 7	8 0 8 0 ::0]	9 0 1 1 9 0 0	0 Rs2 10 0	11 0	1 12 0	13 0 4 13 0 0	14 1 14 0	Rs1 15 0 Hexa Rs1 15 15 15 15 15 15 15 15 15 15 15 15 15	16 0 adeci Beq 16 0 Add	mal (di x9, 17	0x00 0x00 0,for 18 0 0xx00 0xfor 18 0	1 29 0 1484 0 0 unct 29 0	3 20 0 4493 3 20 0	21 0 4 21 0 4	1 Imm 22 1	23 0 1 [4:0 23 0	0 0,11] 24 0	25 1 9	26 1 26	27 1 27	28 1 0 28 0	29 0 3 3 0 0 29 0 3	30 0 de 30 0	31 1	32 1
1 0 0 1 1 F	0 Ir 2 1	mm 3 1	[11:4 1	1 5 0 0 0 5 1 E	6 0 1 mm 6	7 0 0 7 1	8 0 8 0 ::0]	9 0 1 9 0 0	Rs2 10	11 0	12 0	13 0 4 13 0 0	14 1 14 0	Rs1 15 0 Hexa	16 0 Beq 16 0 Addeci	17 1 8 mal ( x0, x	0x00 0x00 0x00 0xfor 18 0 0xFE x18,	1 unct 29 0 1484 0 unct 29 0	3 20 0 4493 20 0	21 0 4 21 0 4	Imm 22 1	23 0 1 [4:0 23 0	0 0,11] 24 0	25 1 9 25 1 E	26	27	28 1 C 28 0	29 0 3 3 0 29 0 29 0 3 3	30 0 de 30 0	31 1	32
1 0 0 1 1 F	0   Ir   2   1	mm 3 1	0 [11: 4 1	I 5 0 0 0 5 1 E	6 0 1 mm 6	7 0 7 1 1 [11 7	8 0 8 0 ::0]	9 0 1 1 9 0 0	0 Rs2 10 0	11 0	1 12 0	13 0 4 13 0 0	14 1 14 0	Rs1 15 0 Hexa	16 0 Beq 16 0 Add 16 1	mal (di x9)  17  1  8  mal (x0, x)  17  0  mal (x10, x10, x10)  17  0  0	0x00 0x00 0xfor 18 0 0x5 0x5 18 0 0x5 18 0	1 unct 29 0 1484 7 0 unct 29 0 0004 7, 0 unct 29	3 20 0 4493 20 0 4E3	21 0 4 21 0 4	1 Imm 22 1	23 0 1 [4:0 23 0	0 0,11] 24 0	25 1 9	26 1 26	27 1 27	28 1 0 28 0	29 0 3 3 0 0 29 0 3	30 0 de 30 0	31 1	32 1
1 0 0 1 1 F	0   Ir   2   1	mm 3 1	0 [11: 4 1	1 5 0 0 0 5 1 E	6 0 1 mm 6	7 0 7 1 1 [11 7	8 0 8 0 ::0]	9 0 1 9 0 0	0 Rs2 10 0	11 0	1 12 0	13 0 4 13 0 0	14 1 14 0	Rs1 15 0 Hexa	16 0 Beq 16 0 Indeed In	mal (di x9, 17 1 8 mal (x0, x 17 0 0 mal (x10, 17 0 0 0 mal (x10, 17 0 0 0 0 mal (x10, 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0x00 0x00 0x00 0x00 0x00 0x00 0x5E x18, F	1 unct 29 0 1484 0 unct 29 0 0000 0,0 unct 29 0	3 20 0 4493 20 0 4E3	21 0 4 21 0 4	1 Imm 22 1	23 0 1 [4:0 23 0	0 0,11] 24 0	25 1 9 25 1 E	26 1 26	27 1 27	28 1 0 28 0	29 0 3 3 0 29 0 29 0 3 3	30 0 de 30 0	31 1	32 1
1 0 0 1 1 F	0   Ir   2   1	mm 3 1	0 [11: 4 1	I 5 0 0 0 5 1 E 5 0 0	6 0 1 mmm 6 0	7 0 7 1 1 7 0 0	8 0 8 0 ::0] 8 0	9 0 1 9 0 0	0 Rs2 10 0	11 0	1 12 0	13 0 4 13 0 0	14 1 14 0	Rs1 15 0 Hexa Rs1 15 0 Hexa Rs1 15 0 Hexa	16 0 Beq 16 0 Indeed In	mal (di x9)  17  1  8  mal (x0, x)  17  0  mal (x10, x10, x10)  17  0  0	0x00 0x00 0x00 0x00 0x00 0x5E 18 0 0x00 0x00 0x00 0x00 0x00	1 unct 29 0 1484 7 0 unct 29 0 0004 7 0 unct 29 0	3 20 0 493 3 20 0 4E3 3 20 0	21 0 4 21 0 4	1 Imm 22 1	23 0 1 [4:0 23 0	0 0,11] 24 0	25 1 9 25 1 E	26 1 26	27 1 27	28 1 0 28 0	29 0 3 3 0 29 0 3 3	30 0 de 30 0	31 1	32 1
1 0 0 1 1 F	0   Irr 2   1   2   0	mm 3 1	[11: 4 1 0	I   5   0   0	6 0 1 mmm 6 0	7 0 7 1 1 [11 7 0	8 0 8 0 :0] 8 0	9 0 1 9 0 0	Rs2 10 0	11 0	12 0	13 0 4 13 0 0	14 1 14 0	Rs1 15 0 Hexa Rs1 15 0 Hexa Rs1 Rs1 Rs1 Rs1 Rs1 Rs1	16 0 Beq 16 0 Add 16 1 Iw	mal (di x9, 17 1 8 mal (x0, x 17 0 0 mal (x10, 17 0 0 mal (x8, 1	0x00 0x00 0xfor 18 0 0x5FE x18, 0 0x6FE x18, 0 0x6FE x18, 0	1 unct 29 0 1484 10 0 0 0 0 0 0 0 0 0 0 0 0 0	3 20 0 4493 20 0 4E3 20 0	21 0 4 21 0 4 21 0 5	1	23 0 1 [4:0 23 0 rd 23 0	0,11] 24 0	25 1 9 25 1 E	26 1 26 0	27 1 27 0	28 1 0 28 0	0pcoo 29 0 3 3 0pcoo 29 0 3	30 0 de 30 0	31 1	32 1
1 0 0 1 1 F	0   Ir   2   1	mm 3 1	0 [11: 4 1	I 5 0 0 0 5 1 E 5 0 0	6 0 1 mmm 6 0	7 0 7 1 7 1 7 0	8 0 8 0 ::0] 8 0	9 0 1 9 0 0	0 Rs2 10 0	11 0	1 12 0	13 0 4 13 0 0	14 1 14 0	Rs1 15 0 Hexa Rs1 15 0 Hexa Rs1 15 0 Hexa	16 0 Beq 16 0 Add 16 1 Iw	mal (di x9, 17 1 8 mal (x0, x 17 0 0 mal (x10, 17 0 0 0 mal (x10, 17 0 0 0 0 mal (x10, 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0x00 0x00 0x00 0x00 0x00 0x5E 18 0 0x00 0x00 0x00 0x00 0x00	1 unct 29 0 1484 10 0 0 0 0 0 0 0 0 0 0 0 0 0	3 20 0 493 3 20 0 4E3 3 20 0	21 0 4 21 0 4	1 Imm 22 1	23 0 1 [4:0 23 0	0 0,11] 24 0	25 1 9 25 1 E	26 1 26	27 1 27	28 1 0 28 0	0pcoo 29 0 3 3 0pcoo 29 0 3	30 0 de 30 0	31 1	32 1

0				0				С				1				2				4				0				3			
														Hexa	deci	mal	0x00	C124	403												
															lw	x9,	8 (sp	)													
				I	mm	[11	:0]							Rs1			F	unct	3			Ird					0	рсос	de		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	1	0	0	0	0	0	1	1
0				0				8				1				2				4				8				3			
														Hexa	deci	mal	0x00	8124	483												
															lw	x18,	4 (sp	)													
				I	mm	[11	:0]							Rs1			F	unct	3			rd					0	рсос	de		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1
0				0				4				1				2				9				0				3			
														Hexa	deci	mal	0x00	8126	623												
															lw	x1, (	0 (sp	)													
				I	mm	[11	:0]							Rs1			F	unct	3			rd					0	рсос	de		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	1
0				0				0				1				2				0				8				3			
														Hexa	deci	mal	0x00	8126	623												
															Add	li sp,	sp, 1	L6													
				I	mm	[11	:0]							Rs1			F	unct	3			rd						рсос	de		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1
0				1				0				1				0				1				1				3			
														Hexa			0x01		113												
															Jar	l x0,	0 (x1	.)													
				- 1	mm	[11	:0]							Rs1				unct				rd						рсос			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	29	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1
0				0				0				0				8				0				6				7			
														Hexa	deci	mal	0x00	0080	067												
ľ																															

Procedimiento del laboratorio:

Proceso realizado para el diseño del ejercicio.

Para el diseño del procesador monociclo RISC-V se implementó teniendo en cuenta las instrucciones suministradas y componentes que fueron proporcionados en el gráfico de dicho procesador, primero se declaraban teniendo en cuenta y agregando como argumentos las variables de entrada y salida, luego se describía el funcionamiento interno de cada módulo.

A continuación se concetaron los modulos y se distanciaron los módulos de cada proceso, además de se asiganaron las instrucciones pertinentes para verificar el funcionamiento correcto de cada módulo y su conexión con módulos posteriores.

Problemas encontrados durante el diseño con sus respectivas las soluciones.

Se encontraron varios problemas en relación al distanciamiento de los módulos y la conexión entre varios componentes, así como algunas implementaciones en las cuales era necesario saber cierta sintaxis del lenguaje para describir el funcionamiento de dichos componentes, consultando en internet y entrando en foros se logró dar con la solución, además de la documentación que fue proporcionada en el Microsft Teams.

Interpretación de los resultados de la simulación.

En la simulación se puede observar la señal de cada uno de los componentes y como se conectaban entre si, para dar lugar al resultado de una operación matemática, se verificó el funcionamiento haciendo manualmente la intrucción y poteriormente se verificaron los resultados

#### Conclusione

RISC es una filosofía de diseño de CPU para computadora que está a favor de conjuntos de instrucciones pequeñas y simples que toman menor tiempo para ejecutarse. El tipo de procesador más comúnmente utilizado en equipos de escritorio, el x86, está basado en CISC en lugar de RISC, aunque las versiones más nuevas traducen instrucciones basadas en CISC x86 a instrucciones más simples basadas en RISC para uso interno antes de su ejecución.

El objetivo de diseñar máquinas la arquitectura RISC FIVE es posibilitar la segmentación y el paralelismo en la ejecución de instrucciones y reducir los accesos a memoria. Las máquinas RISC protagonizan la tendencia actual de construcción de microprocesadores. PowerPC,2 DEC Alpha, MIPS, ARM, SPARC son ejemplos de algunos de ellos